For problems involving dynamic programming, you need only determine the necessary subproblems and their relationship with the smaller subproblems as in class. Furthermore for the dynamic programming problems, you need only determine the amount or length of the optimal answer, not the answer itself. Of course, you must give a running time.

**CLRS Problems**

1. 15-6: Moving on a checkerboard.
2. 16-1: Coin changing. For a, b, and c, prove carefully that your greedy algorithm is correct, as done for the activity selection problem

**Problem 3: Palindromic subsequences**

A subsequence is palindromic if it is the same whether read right to left or left to right. For instance, the sequence A,C,G,T,C,A,G,A has many palindromic subsequences, AAA, ACCA, GTG, (however ACGCGA is NOT palindromic). Devise an efficient algorithm that takes a sequence $A[1\ldots n]$ and returns the length of the longest palindromic sequence. The running time should be $O(n^2)$.

**Problem 4: Updating minimum spanning trees**

You are given a graph $G = (V, E)$ with positive edge weights and a minimum spanning tree for $G$, $T = (V, E')$ with respect to the given weights. Now suppose that the weight of a single edge, $e$ is decreased. We wish to quickly update the MST, without actually recomputing the entire tree from scratch. Give a linear-time algorithm for updating the tree. (Note: There are two cases, when the updated edge is part of the MST and when the edge is not part the MST.) Hint: Don’t forget the cut property.

**Problem 5: Feedback edge sets**

A feedback edge set of an undirected graph $G = (V, E)$ is a subset of edges $E' \subset E$ that intersects every cycle of the graph. Thus, removing the edges of $E'$ will render the graph acyclic. Give an efficient algorithm for the following: Given an undirected graph $G = (V, E)$ with positive edge weights, $w_e$, output a feedback edge set $E'$ of minimum total weight $\sum_{e \in E'} w_e$.

---

1 You may work with a partner, but you must submit your own solutions. You may NOT consult any textbook other than CLRS nor the internet to aid in solving these problems.
Problem 6: Invasions

Two armies simultaneously invade a country. Let’s call them the “red army” and the “blue army.” The red army starts out occupying city \( a \), and the blue army starts out occupying city \( b \). Both armies fan out simultaneously in all directions, and whichever army arrives at a city first, occupies that city, and blocks the other army from either occupying or transiting through that city. The occupying army leaves a small occupation force at that city, but the remainder of the army continues to fan out to all neighboring cities. *In case of a tie, the city is occupied by neither army, and neither army may transit the city.* The army that occupies the most cities wins the war. The question is: which army wins? Let’s model this problem as a directed graph with positive edge weights. The nodes in the graph represent the cities, and the edges represent roads between cities. The weight of an edge \((u, v)\) represents the amount of time required for either army to travel from city \( u \) to city \( v \). Design and analyze an efficient algorithm to solve this problem. The input is a directed, weighted graph, along with distinct nodes \( a \) and \( b \). The output is “red wins,” “blue wins,” or “tie.”

Problem 7: Restricted Single Source shortest paths

Suppose you are given a graph \( G \) with positive edge weights, a node \( s \) and a positive integer \( k \). Give an algorithm which finds the shortest path from \( s \) to all nodes in \( G \), using at most \( k \) edges. (Hint: Either A) Consider modifying the “update” procedure from Dijkstra’s algorithm to use dynamic programming or B) Consider using a graph cloning technique).